TRUSTED® COMPUTING GROUP

SPECIFICATION

Storage Component Class Registry

Version 1.0
Revision 16
June 27, 2022

Contact: admin@trustedcomputinggroup.org

PUBLIC REVIEW

Work in Progress

This document is an intermediate draft for comment only and is subject to change without notice. Readers should not design products based on this document.

DISCLAIMERS, NOTICES, AND LICENSE TERMS

THIS SPECIFICATION IS PROVIDED “AS IS” WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

Without limitation, TCG disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification and to the implementation of this specification, and TCG disclaims all liability for cost of procurement of substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, direct, indirect, or special damages, whether under contract, tort, warranty or otherwise, arising in any way out of use or reliance upon this specification or any information herein.

Contact the Trusted Computing Group at www.trustedcomputinggroup.org for information on specification licensing through membership agreements.

CHANGE HISTORY

REVISION	DATE	DESCRIPTION
Version 1.0, Revision 16	June 27, 2022	• Release of Version 1.0's draft for public review.

CONTENTS

1 SCOPE

This specification is a registry that maps component identifiers within ATA [3], SCSI [4] and NVM Express (NVMe®) [5] capabilities into a Platform Certificate that complies with the specification TCG Platform Certificate Profile version 1.1 or later [1] and specifies the values required in a Platform Certificate that claims usage of this registry.

1.1 Purpose

This specification defines the mapping of data from ATA, SCSI and NVMe capabilities, specifically how the data is encoded within the Platform Certificate. This mapping in this specification enables scalable component identification via ATA, SCSI and NVMe capabilities and verification using the Platform Certificate.

1.2 Key Words

The key words “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” in this document’s normative statements are to be interpreted as described in RFC-2119, Key words for use in RFCs to Indicate Requirement Levels.

1.3 Statement Type

Please note a very important distinction between different sections of text throughout this document. There are two distinctive kinds of text: informative comment and normative statements. Because most of the text in this specification will be of the kind normative statements, the authors have informally defined it as the default and, as such, have specifically called out text of the kind informative comment. They have done this by flagging the beginning and end of each informative comment and highlighting its text in gray. This means that unless text is specifically marked as of the kind informative comment, it can be considered a kind of normative statements.

EXAMPLE: Start of informative comment

This is the first paragraph of 1–n paragraphs containing text of the kind *informative comment* ...

This is the second paragraph of text of the kind *informative comment* ...

This is the nth paragraph of text of the kind *informative comment* ...

To understand the TCG specification the user must read the specification. (This use of MUST does not require any action).

End of informative comment

2 Definitions

2.1 Platform Certificate `componentIdentifier` field

Start of informative comment

A Platform Certificate's **`componentIdentifier`** field is defined in [1]. Each **`componentIdentifier`** in a Platform Certificate represents an individual component in the platform. The **`componentIdentifier`** field is encoded as an ASN.1 SEQUENCE. The fields inside the sequence record data about the individual component.

This specification documents the mapping of ATA [3], SCSI [4] and NVMe [5] components' data and encapsulation of that mapped data within the **`componentIdentifier`** field.

There are other registries than specified in this specification. This specification does not specify that a Platform Certificate issuer is required to use the registry specified in this specification. The specific registry used to translate and encapsulate a specific ATA, SCSI or NVMe component's information is chosen on a component-by-component basis by the Platform Certificate issuer.

End of informative comment

2.1.1 The `componentClassRegistry` field

Start of informative comment

This specification does not specify the inclusion of any optional field within the **`componentIdentifier`**.

If a Platform Certificate issuer wants to encode storage information differently from the recommendation of this specification, they should use the TCG Component Class Registry, identified by the OID **`tcg-registry-componentClass-tcg`**, which is specified in [1].

End of informative comment

The **`OBJECT IDENTIFIER tcg-registry-componentClass-storage`** SHALL be used in the **`componentClassRegistry`** field to identify the registry defined by this specification.

If the registry defined by this specification is identified via the **`componentClassRegistry`** field of a **`componentIdentifier`**, the encoding defined in this specification SHALL be used for the specified **`componentIdentifier`**'s fields.

2.1.2 ASN.1 definition of `tcg-registry-componentClass-storage`

Start of informative comment

`tcg-registry-componentClass-storage` is a TCG-defined OID that is part of the OID namespace inherited from TCPA specifications and [1]. For the convenience of the Reader, the OIDs used by this specification are:

```
-- TCG specific OIDs
tcg OBJECT IDENTIFIER ::= {
   joint-iso-itu-t(2) international-organizations(23) tcg(133) }
tcg-registry OBJECT IDENTIFIER ::= {tcg 18}

-- TCG Registry OIDs
tcg-registry-componentClass OBJECT IDENTIFIER ::= {tcg-registry 3}
```

End of informative comment

```
tcg-registry-componentClass-storage OBJECT IDENTIFIER ::= {
                                    tcg-registry-componentClass 5 }
```


2.2 Notation

Start of informative comment

ATA, SCSI and NVMe capabilities and fields may contain information that is relevant to a **`componentIdentifier`**. This section defines notation used later in this specification to read data from ATA, SCSI and NVMe capabilities and fields, and to format that data into a **`componentIdentifier`**.

This specification uses the “0x” prefix for hexadecimal notation. Any number without the prefix should be interpreted as decimal.

Refer to [3] for the alignment of ATA strings. The Reader is reminded that ATA swaps each pair of bytes in a string.

Refer to [4] for the alignment of SCSI strings. The Reader is reminded that SCSI uses both left-aligned and right-aligned ASCII strings.

Refer to [5] for endianness of NVMe values. The Reader is reminded that NVMe values assume a little-endian ordering convention, unless explicitly specified otherwise. When the registry defined by this specification is used, NVMe values are encoded using their big-endian hexadecimal representation.

Refer to [6] for endianness of PCIe Configuration Space values. The Reader is reminded that PCIe Configuration Space values assume a little-endian ordering convention, unless explicitly specified otherwise. When the registry defined by this specification is used, PCIe register values are encoded using their big-endian hexadecimal representation.

Users of this specification, particularly implementers of a Verifier, need to be aware of the different endianness used in TCG specifications related to PCIe components, which include NVMe devices. For example, the TCG PC Client Specific Platform Firmware Profile Specification [2] does not specify endianness for PCIe device context in measurement, allowing little-endian to be used.

When a value or field does not exist or is not returned, an empty UTF8 string is used with the delimiters.

End of informative comment

The UTF8 encoding characters of a value MUST be the big-endian hexadecimal representation and MUST use uppercase letters.

2.2.1 Concatenation

Start of informative comment

This specification uses different parts that can be included in the same UTF8String; the vertical pipe character (“|”) is used to denote concatenation of two strings.

End of informative comment

The notation “|” means that the value on either side of the vertical pipe character MUST be concatenated.

2.2.2 Literal Value

Start of informative comment

This specification uses big-endian values that are encoded in byte arrays. Each literal value needs to be encoded on a single byte.

End of informative comment

When a cell from column Content in Table 1, Table 2 or Table 3 contains the hexadecimal representation of a value, that value MUST be used, encoded in a single byte.

2.2.3 Literal String

Start of informative comment

This specification uses colons (":") to delimit different fields that are included in the same UTF8String.

End of informative comment

When a cell from column Content in Table 1, Table 2 or Table 3 contains a string inside quotation marks, that string MUST be used.

2.2.4 ATA_FormFactor()

Start of informative comment

The ATA IDENTIFY DEVICE data log contains the *NOMINAL FORM FACTOR* field, which indicates the dimensional or connector form factor of the device. The *NOMINAL FORM FACTOR* field's value is encoded in one byte in a Platform Certificate using this specification.

End of informative comment

When a cell from the Content column in Table 1 contains the notation ATA_FormFactor(), the value of *NOMINAL FORM FACTOR* field of the IDENTITY DEVICE data log MUST be used.

2.2.5 ATA_AOI()

Start of informative comment

The ATA IDENTIFY DEVICE data log contains the *IEEE AOI* field inside the *WORLD WIDE NAME* field.

This specification uses the IEEE 24-bit Administered Organizational Identifier (AOI) defined in [9]. An AOI is encoded in a UTF8String using the base-16 form defined in [3]: the octet array 0xACDE48234567019F is represented by the string "ACDE48234567019F".

End of informative comment

When a cell from the Content column in Table 1 contains the notation ATA_AOI(), the UTF8 characters encoding the AOI base-16 form of the *IEEE AOI* field of the IDENTITY DEVICE data log MUST be used.

2.2.6 ATA_UNIQUEID()

Start of informative comment

The ATA IDENTIFY DEVICE data log contains the *UNIQUE ID* field inside the *WORLD WIDE NAME* field.

End of informative comment

When a cell from the Content column in Table 1 contains the notation ATA_UNIQUEID(), the UTF8 characters encoding the *UNIQUE ID* field of the IDENTITY DEVICE data log MUST be used.

2.2.7 ATA_String(*field)*

Start of informative comment

Some *fields* of the ATA IDENTITY DEVICE data log are already encoded in ASCII printable characters (code values between 0x20 and 0x7E). When encoding such a *field*, all ASCII padding space characters (0x20) are omitted.

End of informative comment

When a cell from the Content column in Table 1 contains the notation ATA_String(*field*), the IDENTITY DEVICE data log's *field* ASCII characters, excluding the padding ASCII space characters, MUST be used.

2.2.8 SPC_INQUIRY_Class()

Start of informative comment

The SCSI Primary Commands (SPC) standard INQUIRY data contains the *PERIPHERAL QUALIFIER* and *PERIPHERAL DEVICE TYPE* fields, which identify the logical unit. The *PERIPHERAL QUALIFIER* and *PERIPHERAL DEVICE TYPE* fields are encoded together in a Platform Certificate using this specification.

End of informative comment

When a cell from the Content column in Table 2 contains the notation SPC_INQUIRY_Class(), the value of Byte 0 of the standard INQUIRY data MUST be used.

2.2.9 SPC_INQUIRY_String(*field*)

Start of informative comment

Some SPC standard INQUIRY data's *fields* are already encoded in ASCII printable characters (code values between 0x20 and 0x7E) with potentially one or more ASCII null character (0x00). Those *fields* are left-aligned as specified in [4]. When encoding such a *field*, all ASCII characters are included in the Platform Certificate.

End of informative comment

When a cell from the Content column in Table 2 contains the notation SPC_INQUIRY_String(*field*), the standard INQUIRY data *field*'s ASCII characters MUST be used.

2.2.10 SPC_VPD_SN_String(*field*)

Start of informative comment

Some SCSI Vital Product Data (VPD) tables' *fields* are already encoded in ASCII printable characters (code values between 0x20 and 0x7E). Those *fields* are right-aligned as specified in [4]. When encoding such a *field*, all ASCII characters are included in the Platform Certificate.

End of informative comment

When a cell from the Content column in Table 2 contains the notation SPC_VPD_SN_String(*field*), the Unit Serial Number VPD page *field*'s ASCII characters MUST be used.

2.2.11 SPC_VPD_DI_UNIQUEID_String()

Start of informative comment

The SCSI Device Identification VPD page, defined in Section 7.7.6 Device Identification VPD page of [4], can contain unique identifiers in some of its *DESIGNATOR* fields. The *DESIGNATOR* field to use, if any is suitable, is determined based on the algorithm detailed below. A Platform Certificate issuer or verifier that follows this specification needs to find the appropriate designation descriptor in the list contained in the SCSI Device Identification VPD page.

The T10 vendor ID based *designator descriptor*, whose *DESIGNATOR TYPE* is 1, is defined in Section 7.7.6.4 T10 vendor ID based designator format of [4]. The *VENDOR SPECIFIC IDENTIFIER* field is a variable length.

The EUI-64 based *designator descriptor*, whose *DESIGNATOR TYPE* is 2, is defined in Section 7.7.6.5 EUI-64 based designator format of [4]. Regardless of the EUI-64 length and format, the *VENDOR SPECIFIC EXTENSION IDENTIFIER* field is a 40-bit numeric value.

The Network Address Authority (NAA) based *designator descriptor*, whose *DESIGNATOR TYPE* is 3, is defined in Section 7.7.6.6 NAA designator format of [4]. For IEEE Registered Extended designator (*NAA* field is 6), the *VENDOR SPECIFIC IDENTIFIER* field concatenated with the *VENDOR SPECIFIC EXTENSION IDENTIFIER* field is a 100-bit numeric value. For IEEE Registered designator (*NAA* field is 5), the *VENDOR SPECIFIC IDENTIFIER* field is a 36-bit numeric value. For IEEE Extended designator (*NAA* field is 3), the *VENDOR SPECIFIC IDENTIFIER A* field concatenated with the *VENDOR SPECIFIC IDENTIFIER B* field is a 36-bit numeric value. For Locally Assigned designator (*NAA* field is 2), the *LOCALLY ADMINISTERED VALUE* field is a 60-bit numeric value.

End of informative comment

When a cell from the Content column in Table 2 contains the notation SPC_VPD_DI_UNIQUEID_String(), the UTF8 characters specified by the algorithm below MUST be used.

Only the *Designator descriptors* whose *ASSOCIATION* field is 0 MUST be used.

By order of preference, the *Designator descriptors* whose *DESIGNATOR TYPE* field is 3, 2 or 1 MUST be used.

1. If a *Designator descriptor* whose DESIGNATOR TYPE field is 3, by order of preference, the *Designator descriptor* whose NAA field is 6, 5, 2 or 3 MUST be used.
    a. If a *Designator descriptor* whose *DESIGNATOR TYPE* field is 3 and *NAA* field is 6, the UTF8 characters encoding the value of the *VENDOR SPECIFIC IDENTIFIER* field concatenated with the *VENDOR SPECIFIC EXTENSION IDENTIFIER* field MUST be used, or
    b. If a *Designator descriptor* whose *DESIGNATOR TYPE* field is 3 and *NAA* field is 5, the UTF8 characters encoding the value of the *VENDOR SPECIFIC IDENTIFIER* field MUST be used, or
    c. If a *Designator descriptor* whose *DESIGNATOR TYPE* field is 3 and *NAA* field is 2, the UTF8 characters encoding the value of the *VENDOR SPECIFIC IDENTIFIER A* field concatenated with the *VENDOR SPECIFIC IDENTIFIER B* field MUST be used, or,
    d. If a *Designator descriptor* whose *DESIGNATOR TYPE* field is 3 and *NAA* field is 3, the UTF8 characters encoding the value of the *LOCALLY ADMINISTERED VALUE* field MUST be used.
2. If a *Designator descriptor* whose *DESIGNATOR TYPE* field is 2, the UTF8 characters encoding the value of the *VENDOR SPECIFIC EXTENSION IDENTIFIER* field MUST be used.
3. If a *Designator descriptor* whose *DESIGNATOR TYPE* field is 1, the *VENDOR SPECIFIC IDENTIFIER* field's ASCII characters MUST be used.
4. If no *Designator descriptor* is suitable, an empty UTF8 string MUST be used.

2.2.12 PCIe_CC()

Start of informative comment

When encoding the PCIe® *Class Code Register*'s value in an OCTET STRING, the 24 bits big-endian representation is used.

End of informative comment

When a cell from the Content column in Table 3 contains the notation PCIe_CC(), the big-endian representation of the PCIe *Class Code register* value MUST be used. If the PCIe *Class Code register* does not exist for the component, the value 0x000000 MUST be used.

2.2.13 NVMe_Val(*Capability or Feature*)

Start of informative comment

When encoding a NVMe Controller or Namespace Structure (CNS) *Capability or Feature*'s value in a UTF8String, the big-endian hexadecimal representation does not include any prefix or suffix. For example, the "0x" prefix is not included in the UTF8String encoding. The UTF8String uses uppercase letters.

End of informative comment

When a cell from the Content column in Table 3 contains the notation NVMe_Val(*Capability or Feature*), the UTF8 characters encoding the value of *register* MUST be used.

2.2.14 NVMe_String(*Capability or Feature*)

Start of informative comment

Some NVMe CNS *Capabilities or Features* are already encoded in ASCII printable characters (code values between 0x20 and 0x7E). Those ASCII strings are left justified as specified in [5]. When encoding such ASCII string, the trailing ASCII space characters (0x20) are omitted.

End of informative comment

When a cell from the Content column in Table 3 contains the notation NVMe_String(*Capability or Feature*), the CNS *Capability or Feature*'s ASCII printable characters, excluding the trailing ASCII space characters, MUST be used.

2.2.15 NVMe_OUI(*Capability or Feature*)

Start of informative comment

This specification uses the IEEE 24-bit OUI defined in [9]. An OUI is encoded in a UTF8String using the base-16 form defined in [9]: the octet array 0xACDE48234567019F is represented by the string "ACDE48234567019F".

The Reader is reminded that the OUI is in little endian format in the Identify Controller Data Structure of NVMe.

End of informative comment

When a cell from the Content column in Table 3 contains the notation NVMe_OUI(*Capability or Feature*), the UTF8 characters encoding the OUI base-16 form of *Capability or Feature* MUST be used.

3 Translation of ATA information into a `componentIdentifier` field

Start of informative comment

ATA does not contain data that is relevant to the **`componentManufacturerId`**, **`fieldReplaceable`**, **`componentAddresses`**, **`componentPlatformCert`**, **`componentPlatformCertUri`**, and **`status`** fields of **`componentIdentifier`**. This specification does not provide any recommendation for the content of those fields.

Table 1 specifies the content of the **`componentClassValue`**, **`componentManufacturer`**, **`componentModel`**, **`componentSerial`** and **`componentRevision`** fields of **`componentIdentifier`**. The **`componentClassValue`** field is encoded as 4 octets, while the remaining fields are encoded as UTF8String in a TCG Platform Certificate and their content is derived from different ATA fields of the IDENTIFY DEVICE data log:

- The *NOMINAL FORM FACTOR* field (page 0x03, offset bytes 32 to 39) of the ATA IDENTIFY DEVICE data log (Log Address 0x30) is defined in Section 9.10.5.5 Form Factor (NOMINAL FORM FACTOR field) of [3]. It is a 4-bit value that is gathered by ATA_FormFactor(), which is the fourth octet of the **`componentClassValue`**.
- The *IEEE AOI* field (page 0x03, offset bits 516 to 539) of the ATA IDENTIFY DEVICE data log (Log Address 0x30) is defined in Section 9.10.5.8 World Wide Name of [3]. It is a 24-bit AOI value assigned by the IEEE Registration Authority, which composes the **`componentManufacturer`** field.
- The *MODEL NUMBER* field (page 0x05, offset bytes 48 to 87) of the ATA IDENTIFY DEVICE data log (Log Address 0x30) is defined in Section 9.10.7.4 MODEL NUMBER field of [3]. It is a 40-byte ASCII string that composes the **`componentModel`** field, since UTF8 is backward compatible with ASCII.
- The *SERIAL NUMBER* field (page 0x05, offset bytes 8 to 27) of the ATA IDENTIFY DEVICE data log (Log Address 0x30) is defined in Section 9.10.7.2 SERIAL NUMBER field of [3]. It is a 20-byte ASCII string that partially composes the **`componentSerial`** field, since UTF8 is backward compatible with ASCII.
- The *UNIQUE ID* field (page 0x03, offset bits 540 to 575) of the ATA IDENTIFY DEVICE data log (Log Address 0x30) is defined in Section 9.10.5.8 World Wide Name of [3]. It is a 36-bit value assigned by the device manufacturer that partially composes the **`componentSerial`** field.
- The *FIRMWARE REVISION* field (page 0x05, offset bytes 32 to 39) of the ATA IDENTIFY DEVICE data log (Log Address 0x30) is defined in Section 9.10.7.3 FIRMWARE REVISION field of [3]. It is an 8-byte ASCII string that composes the **`componentRevision`** field, since UTF8 is backward compatible with ASCII.

End of informative comment

The **componentIdentifier** structure SHALL be populated with values obtained from the methods specified in Table 1's Content column for an ATA-compliant component. The values SHALL be encoded as specified in Table 1.

Table 1: Component Identity Information for ATA (normative)

Field name	ASN.1 Encoding	Content
componentClassValue	OCTET STRING SIZE(4)	0x00 | 0x00 | 0x00 | ATA_FormFactor()
componentManufacturer	UTF8String	ATA_AOI()
componentModel	UTF8String	ATA_String(*MODEL NUMBER*)
componentSerial	UTF8String	ATA_String(*SERIAL NUMBER*) | ":" | ATA_UNIQUEID()
componentRevision	UTF8String	ATA_String(*FIRMWARE REVISION*)

4 Translation of SCSI information into a `componentIdentifier` field

Start of informative comment

The SPC standard does not contain data that is relevant to the **`componentManufacturerId`**, **`fieldReplaceable`**, **`componentAddresses`**, **`componentPlatformCert`**, **`componentPlatformCertUri`**, and **`status`** fields of **`componentIdentifier`**. This specification does not provide any recommendation for the content of those fields.

Table 2 specifies the content of the **`componentClassValue`**, **`componentManufacturer`**, **`componentModel`**, **`componentSerial`** and **`componentRevision`** fields of **`componentIdentifier`**. The **`componentClassValue`** field is encoded as 4 octets, while the remaining fields are encoded as UTF8String in a TCG Platform Certificate, and their content is derived from different SPC fields of the standard INQUIRY data, the Unit Serial Number VPD page, or the Device Identification VPD page:

- The *PERIPHERAL QUALIFIER* and *PERIPHERAL DEVICE TYPE* fields (byte 0) of the SPC standard INQUIRY data are defined in Section 6.7.2 Standard INQUIRY data of [4]. Together, they form an 8-bit value that is gathered by SPC_INQUIRY_Class(), which is the fourth octet of the **`componentClassValue`**.
- The *T10 VENDOR IDENTIFICATION* field (bytes 8 to 15) of the SPC standard INQUIRY data is defined in Section 6.7.2 Standard INQUIRY data of [4]. It is an 8-byte left-aligned ASCII string that composes the **`componentManufacturer`** field, since UTF8 is backward compatible with ASCII.
- The *PRODUCT IDENTIFICATION* field (bytes 16 to 31) of the SPC standard INQUIRY data is defined in Section 6.7.2 Standard INQUIRY data of [4]. It is a 16-byte left-aligned ASCII string that composes the **`componentModel`** field, since UTF8 is backward compatible with ASCII.
- The *DESIGNATOR* field (starting at byte 4) of the SPC Device Identification VPD page (page code 0x83) is defined in Section 7.7.6 Device Identification VPD page of [4]. Some data of the *DESIGNATOR* field can be used to partially compose the **`componentSerial`** field; this is detailed in Section 2.2.11.
- The *PRODUCT SERIAL NUMBER* field (starting at byte 4) of the SPC Unit Serial Number VPD page (page code 0x80) is defined in Section 7.7.19 Unit Serial Number VPD page of [4]. When present, it is a variable length right-aligned ASCII string that partially composes the **`componentSerial`** field, since UTF8 is backward compatible with ASCII.
- The *PRODUCT REVISION LEVEL* field (bytes 32 to 35) of the SPC standard INQUIRY data is defined in Section 6.7.2 Standard INQUIRY data of [4]. A firmware update of the SCSI device could change the value of this field. It is a 4-byte left-aligned ASCII string that composes the **`componentRevision`** field, since UTF8 is backward compatible with ASCII.

End of informative comment

The **componentIdentifier** structure SHALL be populated with values obtained from the methods specified in Table 2's Content column for a SCSI-compliant component. The values SHALL be encoded as specified in Table 2.

Table 2: Component Identity Information for SCSI (normative)

Field name	ASN.1 Encoding	Content
componentClassValue	OCTET STRING SIZE(4)	0x01 | 0x00 | 0x00 | SPC_INQUIRY_Class()
componentManufacturer	UTF8String	SPC_INQUIRY_String(*T10 VENDOR IDENTIFICATION*)
componentModel	UTF8String	SPC_INQUIRY_String(*PRODUCT IDENTIFICATION*)
componentSerial	UTF8String	SPC_VPD_DI_UNIQUEID_String() | ":" | SPC_VPD_SN_String(*PRODUCT SERIAL NUMBER*)
componentRevision	UTF8String	SPC_INQUIRY_String(*PRODUCT REVISION LEVEL*)

5 Translation of NVMe information into a `componentIdentifier` field

Start of informative comment

The NVMe identity data structures (e.g., CNS) do not contain data that is relevant to the **`componentManufacturerId`**, **`fieldReplaceable`**, **`componentAddresses`**, **`componentPlatformCert`**, **`componentPlatformCertUri`**, and **`status`** fields of **`componentIdentifier`**. This specification does not provide any recommendation for the content of those fields.

Table 3 specifies the content of the **`componentClassValue`**, **`componentManufacturer`**, **`componentModel`**, **`componentSerial`** and **`componentRevision`** fields of **`componentIdentifier`**. The **`componentClassValue`** field is encoded as 4 octets, while the remaining fields are encoded as UTF8String in a TCG Platform Certificate and their content is derived from different Controller Capabilities or Features of the CNS:

- The *Class Code Register* is defined in sections 7.5.1.1 Type 0/1 Common Configuration Space, and 7.5.1.1.6 Class Code Register (Offset 09h) of the PCIe specification version 5.0 [6] or 4.0 [7]. For PCIe version 3.0 [8], *Class Code Register* is defined in sections 7.5.1 Type 0/1 Common Configuration Space, 7.5.2 Type 0 Configuration Space Header, and 7.5.3 Type 1 Configuration Space Header. The *Class Code Register* is a 24-bit value, which is converted into an OCTET STRING. For PCIe version 4.0 or 5.0: the second octet of the **`componentClassValue`** contains the ***Base Class Code***, the third octet of the **`componentClassValue`** contains the ***Sub-Class Code*** and the fourth octet of the **`componentClassValue`** contains the ***Programming Interface*** of the *Class Code Register*.
- The *PCI Vendor ID* (*VID*) and *PCI Subsystem Vendor ID* (SS*VID*) are defined in Sections 4.3.1 PCI Vendor ID (VID) and PCI Subsystem Vendor ID (SSVID), and 5.17.2.1 of [5]. They are part of the Identify Controller Data Structure (CNS = 0x01), VID at bytes 0x00 to 0x01 and SSVID at bytes 0x02 to 0x03. Each is a 16-bit value, assigned by the PCI SIG, that partially forms the **`componentManufacturer`** field.
- The *IEEE OUI Identifier* (*IEEE*) is defined in Sections 4.3.3 IEEE OUI Identifier (IEEE), and 5.17.2.1 Identify Controller Data Structure (CNS 01h) of [5]. It is part of the Identify Controller Data Structure (CNS = 0x01), at bytes 0x73 to 0x75. It is an OUI 24-bit value assigned by the IEEE Registration Authority that partially forms the **`componentManufacturer`** field.
- The *Model Number* (*MN*) is defined in Sections 4.3.2 Serial Number (SN) and Model Number (MN), and 5.17.2.1 Identify Controller Data Structure (CNS 01h) of [5]. It is part of the Identify Controller Data Structure (CNS = 0x01), at bytes 0x24 to 0x63. It is a 40-byte left justified (big endian) ASCII string assigned by the component's vendor that forms the **`componentModel`** field.
- The *Field Replaceable Unit (FRU) Globally Unique Identifier* (FGUID) is defined in Sections 4.3.4 IEEE Extended Unique Identifier (EUI64), and 5.17.2.1 Identify Controller Data Structure (CNS 01h) of [5]. It is part of the Identify Controller Data Structure (CNS = 0x01), at bytes 0x112 to 0x127. It is a 16-byte big endian value containing a EUI64 and an 8 bytes extension identifier assigned by the vendor that partially forms the **`componentSerial`** field. The OUI Identifier within the EUI64 is in big endian.
- The *Serial Number* (*SN*) is defined in Sections 4.3.2 Serial Number (SN) and Model Number (MN), and 5.17.2.1 Identify Controller Data Structure (CNS 01h) of [5]. It is part of the Identify Controller Data Structure (CNS = 0x01), at bytes 0x04 to 0x23. It is a 20-byte left justified (big endian) ASCII string assigned by the component's vendor that partially forms the **`componentSerial`** field.
- The V*ersion* (*VER*) is defined in Sections 3.1.3.2 Offset 8h: VS – Version, and 5.17.2.1 Identify Controller Data Structure (CNS 01h) of [5]. It is part of the Identify Controller Data Structure (CNS = 0x01), at bytes 0x80 to 0x83.

It is a 4-byte value containing the major, minor and tertiary version of the NVMe base specification that forms the **`componentRevision`** field.

- The *Firmware Revision* (*FR*) is defined in Sections 5.16.1.4 Firmware Slot Information (Log Identifier 03h), and 5.17.2.1 Identify Controller Data Structure (CNS 01h) of [5]. It is part of the Identify Controller Data Structure (CNS = 0x01), at bytes 0x64 to 0x71. It is an 8-byte ASCII string.
- The NVM Subsystem NVMe Qualified Name (SUBNQN) is defined in Sections 4.5 NVMe Qualified Names, and 5.17.2.1 Identify Controller Data Structure (CNS 01h) of [5]. It is part of the Identify Controller Data Structure (CNS = 0x01), at bytes 0x300 to 0x3FF. It is a UTF-8 string that partially composes the **`componentSerial`** field.

End of informative comment

The **componentIdentifier** structure SHALL be populated with values obtained from the methods specified in Table 3's Content column for a NVMe-compliant component. The values SHALL be encoded as specified in Table 3.

Table 3: Component Identity Information for NVMe (normative)

Field name	ASN.1 Encoding	Content
componentClassValue	OCTET STRING SIZE(4)	0x02 | PCIe_CC()
componentManufacturer	UTF8String	NVMe_Val(*VID*) | ":" | NVMe_Val(*SSVID*) | ":" | NVMe_OUI(*IEEE*)
componentModel	UTF8String	NVMe_String(*MN*)
componentSerial	UTF8String	NVMe_Val(*FGUID*) | ":" | NVMe_String(*SN*) | ":" | NVMe_String(SUBNQN)
componentRevision	UTF8String	NVMe_Val(*VER*) | ":" | NVMe_String (*FR*)

6 References

[1] TCG Platform Certificate Profile Version 1.1 Revision 19 or later. Infrastructure Working Group, TCG. https://trustedcomputinggroup.org/resource/tcg-platform-certificate-profile/

[2] TCG PC Client Specific Platform Firmware Profile Specification. PC Client Working Group, TCG. https://trustedcomputinggroup.org/resource/pc-client-specific-platform-firmware-profile-specification/

[3] Information Technology - ATA Command Set - 5 (ACS-5), INCITS 558-2021, ANSI. https://webstore.ansi.org/Standards/INCITS/INCITS5582021

[4] Information Technology - SCSI Primary Commands - 5 (SPC-5), INCITS 502-2019, ANSI. https://webstore.ansi.org/Standards/INCITS/INCITS5022019

[5] NVM Express® Base Specification Revision 2.0b, NVM Express, Inc. https://nvmexpress.org/wp-content/uploads/NVM-Express-Base-Specification-2.0b-2021.12.18-Ratified.pdf

[6] PCI Express® Base Specification Revision 5.0, Version 1.0. PCI-SIG. https://members.pcisig.com/wg/PCI-SIG/document/13005

[7] PCI Express® Base Specification Revision 4.0, Version 1.0. PCI-SIG. https://members.pcisig.com/wg/PCI-SIG/document/10912

[8] PCI Express® Base Specification Revision 3.0. PCI-SIG. https://members.pcisig.com/wg/PCI-SIG/document/download/8265

[9] Guidelines for Use of Extended Unique Identifier (EUI), Organizationally Unique Identifier (OUI), and Company ID (CID), IEEE. https://standards.ieee.org/content/dam/ieee-standards/standards/web/documents/tutorials/eui.pdf

[10] PCIe-based Component Class Registry Version 1.0 Revision 18 or later. Infrastructure Working Group, TCG. https://trustedcomputinggroup.org/resource/pcie-based-component-class-registry/

Appendix A: Examples

A.1 Sample `PlatformConfiguration`

Start of informative comment

This sample **`PlatformConfiguration`** sequence only shows the **`componentIdentifiers`** sequence. The first component's information of the **`componentIdentifiers`** sequence is based on the ATA Component Sample 1 presented in section A.2.1, the second component's information is based on the SCSI Component Sample 1 presented in section A.3.1, the third component's information is based on the SCSI Component Sample 2 presented in section A.3.2 and the forth component's information is based on the NVMe Component Sample 1 presented in section A.4.1.

```
  SEQUENCE (1 elem)
    [0] (3 elem)
      SEQUENCE (5 elem)
        SEQUENCE (2 elem)
          OBJECT IDENTIFIER 2.23.133.18.3.5
          OCTET STRING (4 byte) 00000002
        UTF8String ABCDEF
        UTF8String Sample Model Number 1
        [0] (25 byte) Sample Serial 1:112233445
        [1] (8 byte) FW Ver 1
      SEQUENCE (5 elem)
        SEQUENCE (2 elem)
          OBJECT IDENTIFIER 2.23.133.18.3.5
          OCTET STRING (4 byte) 0100007F
        UTF8String EXAMPLE
        UTF8String SCSI Model 1
        [0] (28 byte) :Sample SCSI Serial Number 1
        [1] (4 byte) SV A
      SEQUENCE (5 elem)
        SEQUENCE (2 elem)
          OBJECT IDENTIFIER 2.23.133.18.3.5
          OCTET STRING (4 byte) 0100007F
        UTF8String EXAMPLE
        UTF8String SCSI Model 2
        [0] (43 byte) 123456789ABCDEF:Sample SCSI Serial Number 2
        [1] (4 byte) SV B
      SEQUENCE (5 elem)
        SEQUENCE (2 elem)
          OBJECT IDENTIFIER 2.23.133.18.3.5
          OCTET STRING (4 byte) 02010802
        UTF8String ABCD:1234:ABCDEF
        UTF8String M2
        [0] (89 byte) 1122334455667788ABCDEF99AABBCCDD:SN1:nqn.2014-
08.com.example:nvme:nvm-subsystem-sn-d78432
        [1] (3 byte) FW3
```

End of informative comment

A.2 Sample ATA Log data

Start of informative comment

The following is sample data that was used to inform the encoding of section A.1. The data blobs are base64 encoded.

End of informative comment

A.2.1 ATA Component Sample 1

Supported Capabilities (page 0x03)

Start of informative comment

```
gAAAAAADAAGAAAAAAAAAAAAAAAAAAAAAgAAAAAAAAAAAAAAAAAAAAgAAAAAAAAAAAAAAAAAAAACAAAAAAAAAA
Fq83vESIzRFgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAA=
```

End of informative comment

Strings (page 0x05)

Start of informative comment

```
gAAAAAAFAAFTYW1wbGUgU2VyaWFsIDEgICAgIAAAAABGVyBWZXIgMQAAAAAAAAAAAU2FtcGxlIE1vZGVsIE51b
WJlciAxICAgICAgICAgICAgICAgIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAA=
```

End of informative comment

A.3 Sample SCSI data

Start of informative comment

The following is sample data that was used to inform the encoding of section A.1. The data blobs are base64 encoded.

End of informative comment

A.3.1 SCSI Component Sample 1

SCSI INQUIRY Data

Start of informative comment

fwAHAAAAAABFWEFNUExFIFNDU0kgTW9kZWwgMSAgICBTViBBAAAAAAAAAAAAAAAAAAAAAAAAAAAAABSgFiMXZ
wXAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAA=

End of informative comment

SCSI Unit Serial Number VPD (page 0x80)

Start of informative comment

f4AAG1NhbXBsZSBTQ1NJIFNlcmlhbCBOdW1iZXIgMQ==

End of informative comment

A.3.2 SCSI Component Sample 2

SCSI INQUIRY Data

Start of informative comment

fwAHAAAAAABFWEFNUExFIFNDU0kgTW9kZWwgMiAgICBTViBCAAAAAAAAAAAAAAAAAAAAAAAAAAAAABSgFiMXZ
wXAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAA=

End of informative comment

SCSI Unit Serial Number VPD (page 0x80)

Start of informative comment

f4AAG1NhbXBsZSBTQ1NJIFNlcmlhbCBOdW1iZXIgMg==

End of informative comment

SCSI Device Identification VPD (page 0x83)

Start of informative comment

f4MALgIBAB5FWEFNUExFIFNhbXBsZSBTZXJpYWwgTnVtYmVyIDMBAwAIMSNFZ4mrze8=

End of informative comment

A.4 Sample NVMe Identify Controller data

A.4.1 NVMe Component Sample 1

Start of informative comment

The following is sample data that was used to inform the encoding of section A.1. The data blobs are base64 encoded.

This sample uses the PCIe Type 0 Header from Component Sample 1 in Section A.2 of [5].

End of informative comment

Identify Controller Data Structure

Start of informative comment

zas0ElNOMSAgICAgICAgICAgICAgICAgTTIgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
EZXMyAgICAgAO/NqwAAAAAABAEAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABEiM0RVZneIq83vmaq7zN
0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAbnFuLjIwMTQtMDguY29tLmV4YW1wbGU6bnZtZTpudm0tc3Vic3lzdGVtLXNuLWQ3ODQzMgAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAA==

End of informative comment

